A Framework to Generate HLA compliant Visualization Federates through Web Services and X3D

Regina B. Araujo¹, Azzedine Boukerche², Ednaldo Pizzolato¹, Fabio M. Iwasaki¹
¹CS, Federal University of S. Carlos, SP, Brazil
{regina}, {ednaldo}, {fmiwasaki}@dc.ufscar.br
²SITE, University of Ottawa, Ottawa, Canada
boukerch@site.uottawa.ca

Abstract
This paper presents an open standard and web-based framework that generates HLA compliant simulation visualization interfaces with support to control and management functionalities. Differently from other existing approaches, our framework can provide greater flexibility for customization and deployment of HLA-based simulations in different hardware and software platforms.

1. Introduction
The modeling and visualization of simulations, particularly those aimed at training situations, such as emergency actions by Special Forces (e.g., police and fire fighters) are not trivial to build, manage and control. By having control over a simulation, in real-time, a specialist user, such as a fire fighter commander, can add and/or remove existing objects in the simulation, as well as change existing object properties, making the application a flexible environment to train human capabilities on different equipments and situations. Management, on the other hand, can provide important statistics on the performance of humans and equipments in different simulation runs. This paper presents an open standard, web3D-based framework, which can generate visualization interfaces to HLA (High Level Architecture) compliant simulations, with support to control and management functionalities.

Similar approaches to 3D visualization of HLA-based simulations were proposed using Java3D, most of them with their own protocol for communication. However, since the time they were presented, Web Services technology became a better alternative. Web Services use XML formatted messages to exchange data using primarily HTTP and HTTPS and are less prone to security-related problems. Moreover the solutions mentioned above do not deal with simulation control and management but only visualization.

For visualization, instead of using Java3D and having the 3D content being compiled along with the Java application or applet, X3D allows the content to be written in a text file that can be changed without the need to recompile the application. Web Services and X3D standards can provide greater flexibility for the customization of the functionalities of an HLA-based visualization application. Moreover, they can provide simulation deployment in different hardware and software platforms.

The paper is organized as follows: Section 2 gives a brief review on HLA and web services showing how they relate to our solution. X3D is briefly reviewed on section 3. Our framework for simulation visualization, control and management is described in Section 4, followed by Conclusions and References.

2. High Level Architecture and Web Services
The framework presented in this paper is based on the IEEE standard High Level Architecture (HLA) for distributed simulations [1]. HLA-based simulations can interoperate and have its components reused. An HLA compliant simulation is referred to as a federate, and a federation is defined as a set of federates working together. The HLA has three main components: the HLA rules, the HLA federate interface specification and the HLA object model template (OMT). The HLA rules define the responsibilities federates and federations must hold. The interface specification defines the standard services and interfaces of an underlying software architecture, the Runtime Infrastructure (RTI), which supports the data exchange among federates. These interfaces are arranged in 7 service groups: federation management, declaration management, object management, ownership management, time management, data distribution management and support services.

Web Services can be used to implement an architecture that meet the requirements of Service Oriented Architecture, such as interoperability between different systems, clear and unambiguous description language.
and retrieval of services. Web Services and HLA can be used to integrate heterogeneous simulations and international efforts have been made towards the provision of standardized Web Service interfaces independent of the RTI implementation (HLA WSDL API) [2][3][4]. This approach eliminates the need for customized software to route the calls from Web Services and a vendor specific RTI implementation. Even though the underlying work is still the same, it is transparent to federates and reusable by other simulations. Next section describes our solution for the creation of HLA-based simulation visualization for control and management that uses web services and X3D for greater flexibility.

3. HLA Compliant Simulation Visualization

There are three techniques for simulation visualization (with varying degree of integration between the simulation control and its visualization): post-processing, tracking and steering [5]. In post-processing technique, the resulting images from the visualization are generated after the simulation ends. In tracking, it is possible to view the current internal state of an executing simulation but not to change it. In the steering technique, the user has control over the simulation parameters and can change them at run time – this is the technique supported by our framework. In a training simulation, the user's inputs are continuously changing his/her corresponding virtual environment states. To interact with a simulation, it is necessary to define a strategy to further refine its visualization [6]. The visualization of a training simulation can be more or less effective depending on its specification and the user's perception [7]. The user can have a richer experience if the visualization specification can be changed during the simulation execution or customized for the user's level of expertise. For this reason, this framework uses the X3D standard [8], VRML successor, to render the simulated environment and user interface. Next section presents how X3D is used in our framework.

3.1 Using Extensible 3D for Visualization

Extensible 3D (X3D) is an ISO standard for real-time 3D graphics that defines a file format using XML and a run-time environment to be embedded in applications. The use of X3D features in our framework allows a wide variety of customization of the visualization client interface. X3D base components are sets of predefined nodes with some functionality. Those nodes are used to describe 3D objects and their relationships in a scene. Unlike Java3D or vendor-specific visualization software for HLA that must be compiled with the application, X3D is interpreted dynamically from human-readable text files. Also, many X3D content authoring software are being released with the support of well known modeling tools like Blender, for example. In order to create an X3D scene from a HLA simulation, the following base nodes are used to generate a basic X3D file:

- Prototype node: a new node type defined in terms of built-in or other prototype nodes by using the PROTO statement.
- Script node: a node used to program a behavior in the scene.
- Sensor node: a node that detects user inputs.
- Transform node: a node to position an object in the space.
- Inline node: a node that embeds a scene from another X3D file.

A mapping between the HLA object and interaction classes to X3D prototype nodes is performed and described in [9]. The prototype is named after the object or interaction class from the HLA/Federation Object Model - FOM (whose main function is to specify, in a common standardized format, the nature of the data exchanged among federates). Its attributes or parameters are linked from the prototype interface fields into a script node fields. Within the script node, function stubs are generated for the attributes or parameters converted to fields. Each prototype can link to other X3D files localized through URL using the inline node. By linking other files, the scene graph they define can be reused, such as the geometric models, animation scripts, user interface etc. Thus, for the same simulation, the visualization federate can have multiple interfaces with varying graphics level of details and specific interface for different users. The description of the framework is presented in the next section.

4. A Framework to Generate Visualization Federates

Our framework generates visualization federates from a FOM and establishes communication with the RTI through Web Services. The federate is a web application and is run within the context of a web browser. The application architecture is shown in Figure 1, and is divided into three layers:

- The communication layer uses the HLA WSDL API to exchange data with a federation through the RTI services;
The application layer is the link between the communication and the presentation layers (Figure 2) and is generated by the framework based on a FOM; the presentation layer embeds the 3D browser into the federate and presents the data from the federation.

This layered architecture supports different X3D browsers (such as Flux Player [10] or the Xj3D toolkit [11]) and different HLA APIs by defining an implementation-specific middleware for web-based or standalone federate deployment. By embedding the visualization application into a web page, the simulation data can be presented in two ways: In the X3D scene; and using HTML document elements such as tables.

Also, when the application is embedded in an HTML document, the AJAX [12] techniques are used to make Web Service calls and exchange data within the application layers. The application written in Java is an applet that exchanges function calls with JavaScript using the LiveConnect or ActiveX technologies.

### 4.1 Application Layer Functionalities

The framework defines functions stubs that must be implemented to attend to federation-specific requirements a federate must meet, such as subscription and publication of data, synchronization points and time management policies, for example. Some functionality for object discovery and update are default for all classes but this can be changed: when a new object instance is discovered from the federation and notified to the Federate Ambassador (responsible for handling all outgoing information passed from the user simulation to the RTI), the corresponding prototype node is instanced into the scene and its handle and reference are kept in the application layer. The handle and reference are used to find and update the object attributes either in the federation or in the scene.

Interactions can be sent by triggering sensor nodes and object attributes can be changed through script nodes. Listener for field change events from the script nodes are generated in the application layer. Those listeners are callback functions that will request the desired service to the RTI Ambassador to send interactions or update object attributes.

A performance evaluation of the prototype was carried out in two different scenarios (Core 2 Duo 2.4GHz processor with 4GB RAM with a GeForce 7950 GT KO 512MB; Pentium 4 HT 3.2GHz with 2GB RAM with a GeForce 6800 XT 256MB). The simulation was hosted locally for both machines, receiving constant updates on position and other attributes of around 50 simulation-driven objects. In both platforms, the frame rate was kept over 15 FPS with low polygon count models. Further data analysis is needed though the update rates of data sent by each federate varies depending on the simulation model.

Performance will depend mainly on browser implementation, and to some extent, it will depend also on the number of scripts being performed. In case performance falls under acceptable response time, the 3D models being used can be replaced by less complex models. More complex simulations are under implementation to be tested in a LAN environment.
5. Conclusions
This paper presents an HLA-based framework that generates HLA-based visualization interfaces with support to simulation control and management. Differently from other existing approaches, our framework can provide greater flexibility for customization and deployment of HLA-based simulations in different hardware and software platforms, by making use of web services and X3D technologies. A performance evaluation of the prototype was made in two different platforms, achieving a 15 fps rate in a scenario with low complexity geometric models at constant update. The tests show that our solution can be a powerful tool for the generation of visualization federates, in which control and management can be done at simulation run time. A graphic tool is being built to make it easier to use our framework to build visualization federates.

6. Acknowledgements
This work was supported by São Paulo State Research Support Foundation, FAPESP, under Process 2006/00741-7 and CAPES.

7. References